Integrated Guidance Navigation and Control for a Fully Autonomous Indoor UAS

Girish Chowdhary*, D. Michael Sobers, Jr.†, Chintasid Pravitra‡,
Claus Christmann §, Allen Wu‡, Hiroyuki Hashimoto¶,
Chester Ong §, Roshan Kalghatgi ‖ and Eric N. Johnson**
Georgia Institute of Technology, Atlanta, GA, 30332-0152, USA

This paper describes the details of a Quadrotor miniature unmanned aerial system capable of autonomously exploring cluttered indoor areas without relying on any external navigational aids such as GPS. A streamlined Simultaneous Localization and Mapping (SLAM) algorithm is implemented onboard the vehicle to fuse information from a scanning laser range sensor, an inertial measurement unit, and an altitude sonar to provide relative position, velocity, and attitude information. This state information, with a self-generated map, is used to implement a frontier-based exhaustive search of an indoor environment. To ensure the SLAM algorithm has sufficient information to form a reliable solution, the guidance algorithm ensures the vehicle approaches frontier waypoints through a path that remains within sensor range of indoor structures. Along with a detailed description of the system, simulation and hardware testing results are presented.

I. Introduction

Autonomous indoor reconnaissance and surveillance can bring key capabilities allowing soldiers to negotiate cluttered and confined areas without risking human life. This technology can also bring increased capabilities in disaster management and monitoring confined urban spaces. Miniature Unmanned Aerial Systems (M-UAS) are ideal candidates for such missions as they can use three dimensional maneuvers to overcome obstacles that cannot be overcome by ground vehicles. However, significant technological challenges exist in order to ensure reliable operation in such environments. Most current algorithms for Unmanned Aerial System (UAS) guidance, navigation, and control rely heavily on GPS signals,[1–3] and hence are not suitable for indoor navigation where GPS and other radio signals are normally not available. Furthermore, the indoor M-UAS must be sufficiently small in order to successfully maneuver in cluttered indoor environments, consequently limiting the amount of computational and sensory power that can be carried onboard the M-UAS. Finally, the M-UAS should be designed to be expendable due to the dangerous environments it needs to operate in, hence low-cost, low-weight designs need to be explored. These restrictions pose significant technological challenges for the design of reliable M-UAS platforms capable of navigating in cluttered areas in a GPS-denied environment.

This paper describes a M-UAS capable of exploring cluttered indoor areas without relying on external navigational aids such as GPS. As shown in Figure 1, the system consists of an air vehicle, a ground station, and a safety pilot interface for manual flight. The vehicle, referred to as the GTQ, is capable of completely autonomous indoor flight. However, a ground station computer is used to monitor vehicle health and status,

*Research Engineer II, School of Aerospace Engineering
†Major, USAF. Graduate of the School of Aerospace Engineering. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.
‡Graduate of the School of Aerospace Engineering
§Graduate Research Assistant, School of Aerospace Engineering
¶Graduate Student, School of Aerospace Engineering
‖Graduate Student, School of Mechanical Engineering
**Lockheed Martin Professor of Avionics Integration, School of Aerospace Engineering

to view the map and sensor output, and to provide observers with the ability to interact with the vehicle if desired. The GTQ is an off-the-shelf quadrotor platform which is equipped with off-the-shelf avionics and sensor packages, using custom software and interface electronics. An elaborate navigation algorithm was developed that fuses information from a scanning laser range sensor, an inertial measurement unit (IMU), and sonar altitude sensor to form an accurate estimate of the vehicle attitude, velocity, and position relative to indoor structures while simultaneously mapping the environment. A streamlined Simultaneous Localization and Mapping (SLAM) routine is implemented to provide position updates to the navigation software. The SLAM techniques are combined with a compact exploration strategy for autonomously exploring indoor environments. Since the SLAM position estimates rely on the presence of features in the environment, the guidance algorithm is closely integrated with the navigation algorithm to ensure the vehicle maintains a trajectory that keeps it within sensor range of walls and other indoor structures. The control architecture augments a proven baseline proportional-derivative controller with an optional adaptive element that aids in mitigating modeling error and other system uncertainties.

A discussion of the vehicle platform and the vehicle dynamic modeling is presented next in Section II. The avionics system is discussed in Section III. The details of the guidance, navigation, and control algorithms are presented in Sections IV, V, and VI respectively. Simulation and hardware testing results are presented in Section VII, and the paper is concluded in Section VIII.

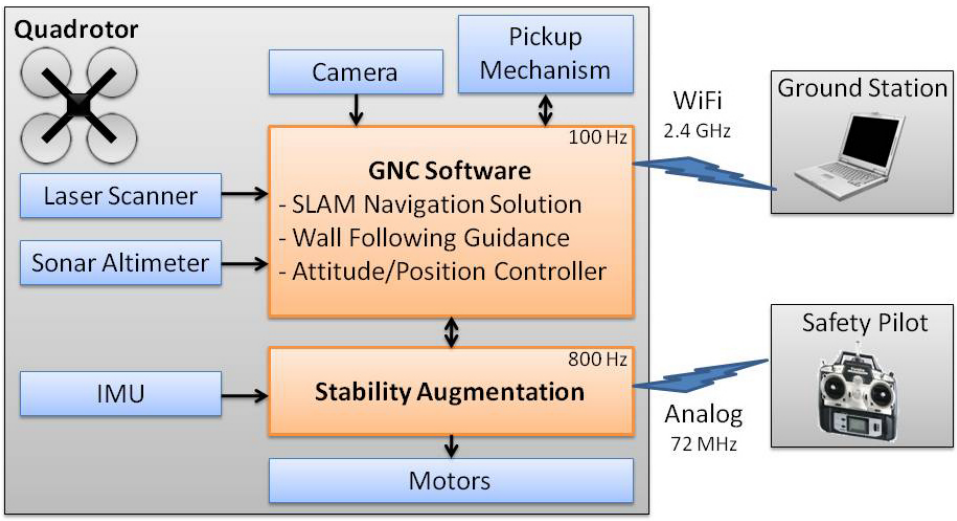
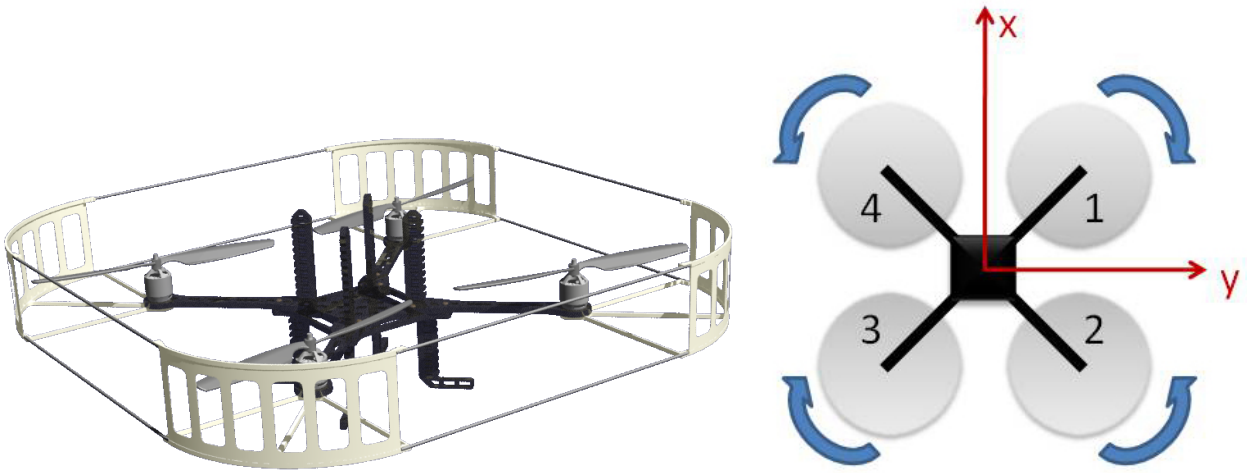


Figure 1. System Architecture. Note: the camera and pickup mechanism are features that enable the GTQ to perform various missions. They are included here for completeness, but are not a necessary part of the GNC algorithms required for indoor flight and are not discussed further in this paper.

II. Description of Vehicle

Quadrotors have become a very popular choice for M-UAS due to their relatively high payload capacity and high maneuverability.[4] Furthermore, unlike helicopters, Quadrotors avoid the use of mechanical parts for exerting moments and forces required for maneuvering.

The AscTec Pelican Quadrotor made by Ascending Technologies GmbH was selected as the base airframe (see Figure 2(a)). The vehicle structure, motors, and rotors of AscTec Pelican were used without modification. The vehicle generates lift using four fixed pitch propellers driven by electric motors. Control is achieved by creating a relative thrust offset between the propellers. Quadrotors can either be flown with diamond configuration (front, right, back, left motors are used to effect pitching, rolling, and yawing motion) or square configuration (front-right, back-right,back-left, front-left motors are used to effect pitching, rolling, and yawing motion). Although many Quadrotors in aerial robotics community fly with diamond configuration,[4–6] the square configuration (shown in Figure 2(b)) was selected to allow for more flexibility in sensor mounting locations.



(a) AscTec Pelican Platform (b) Quadrotor Flying in Square Configuration

Figure 2. The GTQ uses the Ascending Technologies Pelican for the aerial platform.

A. Quadrotor Dynamic Modeling

Quadrotor dynamics flying in configuration shown in Figure 2(b) has been modeled in simulation. Assuming near hover aerodynamics, fuselage aerodynamics and forward flight rotor aerodynamics can be neglected. Hence, the total force acting on the Quadrotor is composed only of thrust and gravity forces. Newton's second law in the body axis can be written as:

$$\begin{bmatrix} 0 & 0 & -(\tau_1+\tau_2+\tau_3+\tau_4) \end{bmatrix}^T + F_g = m\frac{^b dv}{dt} + \omega \times mv \tag{1}$$

where τ_i represents thrust magnitude on the ith rotor for $i = 1, 2, 3, 4$. F_g represents gravity force acting on the vehicle in the body frame, $v \in \Re^3$ is the velocity in the body frame, $\frac{^b dv}{dt}$ is the derivative of the body velocity with respect to the body frame, $\omega \in \Re^3$ is the angular rate of the body, and m represents the mass.

Neglecting forward flight aerodynamics, total moment acting on the Quadrotor composes of four different sources: hub yawing moment (M_{hy}), differential thrust moment (M_{dt}), inertial reaction moment (M_{ir}), and gyroscopic moment (M_{gy}).[4–6] The primary moment contributions are from hub yawing moment and differential thrust moment while inertial reaction moment and gyroscopic moment provides insignificant contribution. Euler's law in the body axis can be written as

$$M = M_{hy} + M_{dt} + M_{ir} + M_{gy} = I\dot{\omega} + \omega \times I\omega \tag{2}$$

where I is the Quadrotor inertia matrix. The individual moment components are defined as follows:

$$M_{hy} = \begin{bmatrix} 0 & 0 & -M_1 + M_2 - M_3 + M_4 \end{bmatrix}^T \tag{3}$$

$$M_{dt} = \begin{bmatrix} l_y(-\tau_1 - \tau_2 + \tau_3 + \tau_4) & l_x(\tau_1 - \tau_2 - \tau_3 + \tau_4) & 0 \end{bmatrix}^T \tag{4}$$

$$M_{ir} = \begin{bmatrix} 0 & 0 & -(I_r\dot{\Omega}_1 - I_r\dot{\Omega}_2 + I_r\dot{\Omega}_3 - I_r\dot{\Omega}_4) \end{bmatrix}^T \tag{5}$$

$$M_{gy} = \begin{bmatrix} -q(I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4) & p(I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4) & 0 \end{bmatrix}^T. \tag{6}$$

where M_i is hub yawing moment magnitude of the i^{th} rotor acting on the body. Ω_i is angular velocity magnitude of the i^{th} rotor. I_r is moment of inertia of the rotor blade. l_x and l_y are distance from center of gravity to rotor hub in x and y directions. p and q are roll and pitch rates written in body axis. The total

moment acting on the body is formed by combining the above terms:

$$M = \begin{bmatrix} l_y(-\tau_1 - \tau_2 + \tau_3 + \tau_4) - q(I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4) \\ l_x(\tau_1 - \tau_2 - \tau_3 + \tau_4) + p(I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4) \\ -M_1 + M_2 - M_3 + M_4 - (I_r\dot{\Omega}_1 - I_r\dot{\Omega}_2 + I_r\dot{\Omega}_3 - I_r\dot{\Omega}_4) \end{bmatrix}. \quad (7)$$

The vehicle's pitching motion can be generated by commanding differential Revolutions Per Minute (RPM) between the two front motors and the two back motors. Rolling motion can be generated by commanding differential RPM between the two right motors and the two left motors. Yawing motion can be generated by increasing motor RPM on one diagonal and reducing motor RPM on the other diagonal.[5] Typical helicopter controls can therefore be mapped to motor commands x_i using Eq.(8).

$$\begin{aligned} x_1 &= \sqrt{\delta_T - \delta_p + \delta_q + \delta_r} & x_2 &= \sqrt{\delta_T - \delta_p - \delta_q + \delta_r} \\ x_3 &= \sqrt{\delta_T + \delta_p - \delta_q - \delta_r} & x_4 &= \sqrt{\delta_T + \delta_p + \delta_q + \delta_r} \end{aligned} \quad (8)$$

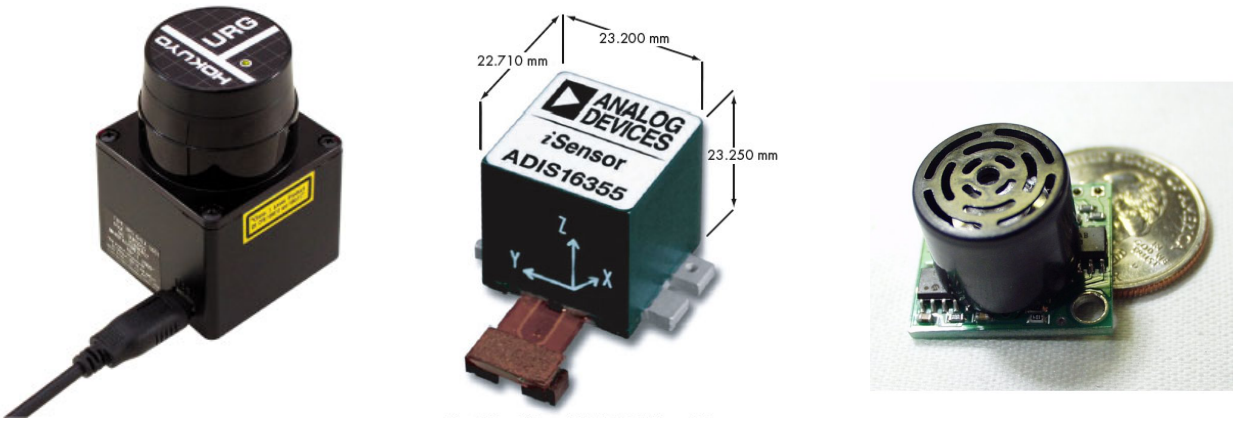
where δ_T, δ_p, δ_q, and δ_r represent thrust, roll, pitch, and yaw commands. In this form the commands are equivalent to a helicopter's collective, lateral cyclic, longitudinal cyclic, and tail rotor commands. A second order model is used to relate motor command x_i to motor RPM Ω_i. The second order model consists of two cascaded first order systems: the first one relates the motor commands to the motor states, and the second one relates the motor states to rotor RPMs. Rotor aerodynamics is modeled using blade element and momentum theory. The relationship between motor command x_i , rotor angular velocity Ω_i, thrust τ_i, and moment M_i is modeled similar to reference [7]. Vehicle's parameters including moments of inertia and maximum achievable RPM are currently being refined through experimentation.

III. Avionics Suite

The UAS uses three primary measurement sensors for navigation, stability and control: a laser range finder, a sonar altimeter, and an inertial measurement unit (IMU). The laser range finder used is the Hokuyo URG-04LX-UG01 (see Figure 3(a)). It is capable of measuring distances up to 4 m and has a maximum detection area of 240 degrees, with a resolution of 1 mm and 0.36 degrees respectively. The sonar altimeter used is the MB1040 LV MaxSonar EZ4 high performance ultrasonic range finder (see Figure 3(c)). It is capable of measuring distances up to 6.45 m away with resolution of 25.4 mm. The IMU is the ADIS-16365-BMLZ built by Analog Devices Inc (see Figure 3(b)). It consists of a three-axis digital gyroscope and three-axis accelerometer that can measure forces up to ± 18 g. These sensors are integrated around the Gumstix Overo Fire onboard computer which is a small and cost effective ARM Cortex-A8 OMAP3530 based computer-on-module. It is equipped with 256 MB Flash RAM, and it can communicate using UART, SPI, and I2C interfaces. The Gumstix Overo computer is equipped with 802.11g and Bluetooth wireless links. Two three-cell Lithium Polymer battery packs are used: one drives the motors to provide lifting power, and the other powers the onboard computer.

IV. Guidance Algorithm

Indoor navigation employing laser aided SLAM is by its nature based on measurements of local features. As a result, any guidance system must use information in the local environment, rather than specifying waypoints fixed in inertial space. This section describes a compact exploration (guidance) strategy that performs an exhaustive search along a path that stays close to walls in an effort to keep the scanning laser rangefinder within range of the features needed for the SLAM algorithm to work. The implementation of SLAM and path-planning algorithms on ground robots is a well-studied problem. However, in extending these methods to M-UASs, designers face significant new challenges. Hovering aircraft have more stringent power and weight constraints, vehicle dynamics are faster and span six degrees of freedom, and air vehicles are typically more delicate and less forgiving of impacts with obstacles in the environment. In fact, many well-known optimal guidance and path planning techniques are not currently feasible for onboard implementation on M-UASs because of limited computational power. As a result, in contrast to many published works on guidance and path planning, the following approach does not focus on optimal solutions for exploration. Instead, it rather emphasizes an efficient algorithm that can quickly determine the location of unexplored



(a) Hokuyo Laser Range Scanner (b) Analog Devices ADIS-16365-BMLZ IMU (c) MaxSonar EZ 4 Sonar altimeter

Figure 3. The GTQ relies on these commercially-available sensors for navigation.

areas while keeping the vehicle within sensor range of geographic features.

SLAM and autonomous exploration for M-UAS application was first demonstrated by Achtelik et. al.[8] Achtelik et. al. uses frontier-based exploration with goal-driven dynamic programming trajectory generation. However, such navigation and guidance algorithms are computationally expensive. Sobers et al. have previously developed a totally self-contained M-UAS architecture with a very compact SLAM algorithm and a simple wall following guidance strategy.[9] However, wall following guidance alone does not place higher weight on unexplored areas. In some cases, the method may only track the outer walls of a building and may leave inner rooms completely unexplored. In other cases, unfavorable geometry may cause the vehicle to stay in the same room or avoid certain rooms altogether. The guidance algorithm described here is designed to improve upon simple wall-following logic by introducing an efficient global book-keeping feature, while keeping the algorithm simple enough to run alongside the SLAM algorithm without over-taxing the onboard flight computer.

Frontier-based exploration can provide a way to "remember" what parts of the map have been previously visited. The method was first introduced by Yamauchi[10] as an effective way for a mobile robot to explore an unknown environment. Without frontier-based exploration, a robot may have to explore an unknown environment randomly with some form of obstacle avoidance logic. The principle of frontier-based exploration is : *"try to get as much new information as possible by going to a boundary between explored and unexplored territory"*. Various forms of frontier-based exploration strategy have been developed, most of which require some form of global map in order to find frontiers and plan trajectories.[10–12] A global map can be grid-based, feature-based, or polygonal-based. However, such global map and its essential guidance algorithms are not computationally practical onboard a M-UAS.

Rather than using a global map, Freda and Oriolo applied the principle of frontier-based exploration to a data structure called Sensor-Based Random Tree (SRT).[13,14] This guidance system uses an SRT method called SRT-Star to store frontiers and safe-regions, and to sequence new waypoints.[13] The SRT-Star metnod is blended with a wall-following algorithm to ensure that the vehicle keeps close to walls, thereby increasing the chance of being in a geometry that is favorable for the SLAM navigation routine.

Assuming near-hover dynamics and a semi-structured environment, first note that the three-dimensional guidance problem can be simplified by decoupling horizontal plane guidance and altitude guidance. Altitude guidance simply commands constant altitude above ground level while horizontal guidance commands horizontal velocity and heading.

The main idea is that the overall velocity command (v_{cmd}) is composed of a contribution from frontier velocity (v_{fr}) and a contribution from wall-following velocity (v_{wf}).

$$v_{cmd} = v_{wf} + v_{fr} \tag{9}$$

The heading command (ψ_{cmd}), on the other hand, is solely generated from the frontier contribution. The raw scan data from the laser rangefinder is used to create an SRT and ultimately compute velocity and

heading commands. The commands are updated upon receipt of new scan data, at a rate of 10 Hz.

Algorithm 1 illustrates the sequence of commands that are executed at a particular update time step.

Algorithm 1 Compute Velocity Command (v_{cmd}) and Heading Command (ψ_{cmd})

Require: x, $x_{waypoint}$, $scan$, SRT, and d
1: $v_{wf} \Leftarrow getWallFollowingVelocity(scan)$
2: **if** $||x - x_{waypoint}|| < d$ **then**
3: $\quad newFrontier \Leftarrow frontierSearch(scan)$
4: $\quad SRT \Leftarrow updateSRT(SRT, newFrontier)$
5: $\quad$ **if** $frontierExist(SRT)$ **then**
6: $\quad\quad x_{waypoint} = newWaypoint(SRT)$
7: $\quad$ **else**
8: $\quad\quad x_{waypoint} = previousWaypoint(SRT)$
9: $\quad$ **end if**
10: **end if**
11: $v_{fr} \Leftarrow getFrontierVelocity(x, x_{waypoint})$
12: $\psi_{cmd} \Leftarrow getHeading(x, x_{waypoint})$
13: $v_{cmd} \Leftarrow v_{fr} + v_{wf}$

The main algorithm requires the vehicle's current position (x), position of commanded waypoint ($x_{waypoint}$), raw scan data ($scan$), sensor based random tree structure (SRT), and threshold distance (d).

The algorithm begins by computing the wall-following velocity component (v_{wf}) directly from the scan data. The purpose of the wall-following velocity is to create an obstacle avoidance potential field while keeping within sensor range of observable features in the environment. Let r_i denote the i^{th} scan range reading shown in Figure 4. Incremental velocity command (v_{ri}) in radial direction due to each scan point is obtained from:

$$v_{ri} = \begin{cases} K_{wf}(r_i - r_t) & \text{if } r_i \geq r_{safe}; \\ K_{safe}(r_i - r_t) & \text{if } r_i < r_{safe}; \end{cases} \tag{10}$$

where K_{wf} and K_{safe} are gains used for nominal and safety cases as determined by safe radius r_{safe}, where r_t is a user specified parameter representing distance from walls that the vehicle tries to maintain.[15]

Let θ_i shown in Figure 4 denote angle of the ith scan with respect to body frame (**xy**), and let n denote number of in-range scans points. Velocity command in body frame u_{cmd} and v_{cmd} results from projecting and summing each incremental velocity.

$$u_{cmd} = \sum_{i=1}^{n} v_{ri} cos\theta_i \qquad v_{cmd} = \sum_{i=1}^{n} v_{ri} sin\theta_i \tag{11}$$

Let ψ be the heading angle referenced from an arbitrary chosen inertial frame (**XY**). Velocity command in body frame can be converted to inertial frame using Equation (12).

$$v_{wf} = \begin{bmatrix} v_{x_{wf}} \\ v_{y_{wf}} \end{bmatrix} = \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} u_{cmd} \\ v_{cmd} \end{bmatrix} \tag{12}$$

The overall effect is attraction to a wall if the vehicle is too far away, and repulsion from a wall if the vehicle is too close. In cases where multiple walls are in sensor range, the resultant velocity depends on wall geometry and range. For instance, the vehicle is attracted to long wall segments located further away more than short wall segments located near by.

The algorithm then checks the distance to the commanded waypoint ($x_{waypoint}$). If the vehicle has not arrived at the commanded waypoint, the waypoint will not be modified. If the vehicle has arrived at the commanded waypoint, a new waypoint is generated from the frontier planner (Algorithm 1, lines 3 through 8). The vehicle is considered to have arrived at the commanded waypoint when it is within distance d of the commanded waypoint. Finally, the commanded waypoint and vehicle's position are used to generate the heading command (ψ_{cmd}) and frontier velocity (v_{fr}). Notice that although commanded waypoint may not change at a particular time step, frontier velocity and heading change at every time step due to changes in the vehicle position.

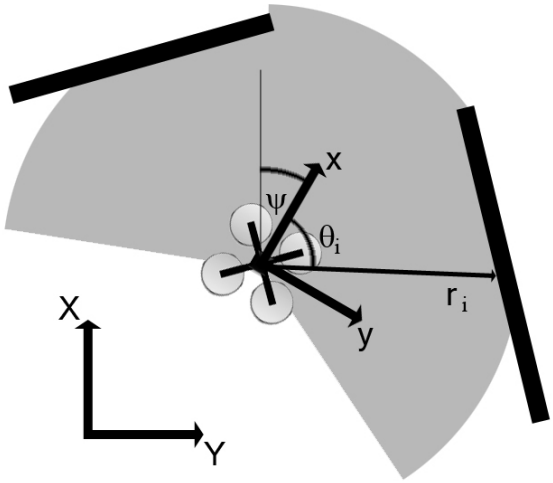


Figure 4. Reference frame description of vehicle and scan points

SRT-Star, as outlined in lines 2 through 10 of Algorithm 1, takes advantage of the 2D laser range sensors scan characteristics. Upon arriving at a waypoint, SRT-Star divides the laser scan data into sectors, where each sector contains a left-point, a mid-point, and a right-point. The mid-point is declared as a frontier if the sector is completely obstacle-free while left-point and right-point are declared as frontiers if there are large discontinuities between adjacent sectors. Sectors and frontier points are illustrated in Figure 5. If at least one frontier exists, a new commanded waypoint is generated by randomizing a point inside a sector that possesses at least one frontier. If no frontier exists, the vehicle backtracks to the previous waypoint.

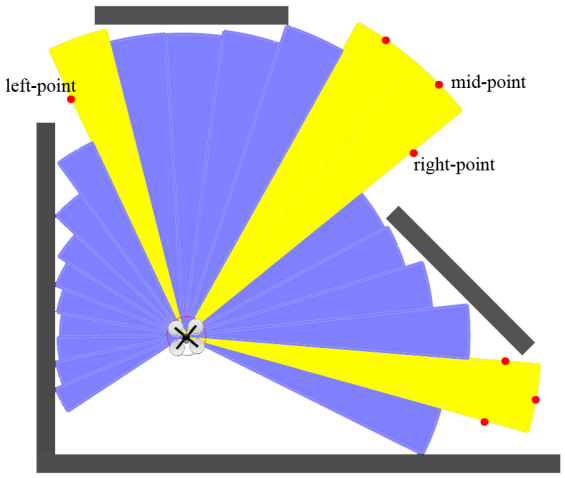


Figure 5. SRT-Star divides laser scan into sectors

Sectors and frontiers are stored in a tree-like structure with waypoints as nodes and frontiers as potential branches. Due to the panning motion of the laser scanner, area within a sector is considered to be free of obstacles. Thus, obstacle free regions also expand as SRT evolves. Expansion of free regions implies that some existing frontiers may fall under newly acquired sectors. In that case, SRT-Star will remove the frontier in order to avoid unnecessary exploration. Details of SRT-Star are discussed in [13].

One difference between the original SRT-Star and our method is that the original SRT-Star feeds the commanded waypoint directly to the controller. The method outlined in Algorithm 1 calculates frontier velocity and heading commands from the commanded waypoint. It then blends the frontier velocity command

with wall following velocity component, and finally feeds velocity and heading commands to the controller.

Let K_{fr} be a frontier gain specified by user. Let $x_{waypoint}$ and $y_{waypoint}$ be the position of the commanded waypoint in the inertial frame as obtained from SRT-Star. Let x and y be the vehicle's current position in the local inertial frame. Given the commanded waypoint, a simple proportional feedback law shown in Equation 13 is used to obtain the frontier velocity component.

$$\vec{v}_{fr} = \begin{bmatrix} v_{x_{fr}} \\ v_{y_{fr}} \end{bmatrix} = \begin{bmatrix} K_{fr}(x_{waypoint} - x) \\ K_{fr}(y_{waypoint} - y) \end{bmatrix} \tag{13}$$

Moreover, the vehicle's heading will always be commanded to point directly toward the commanded waypoint.

$$\psi_{cmd} = atan2(y_{waypoint} - y, x_{waypoint} - x) \tag{14}$$

Note that this heading command also provides another benefit. It steers the scan field-of-view toward the neighborhood of the commanded waypoint. The wall-following velocity component will command attraction to any walls in the vicinity. Therefore, the wall following velocity component also implicitly brings the vehicle towards the commanded waypoint.

To improve safety of the vehicle, the commanded velocity and the yaw rate are limited in the onboard software. Furthermore, a time-out is implemented between any two waypoints. This is particularly useful to counter drift of the inertial frame due to imperfections in the SLAM-based navigation solution, which can result in waypoints placed in inaccessible areas of the map.

V. Navigation Algorithm

Missions that require UAVs to enter unknown environments, including structures, may not have reliable reception of radio signals. As a result, reliance on GPS for navigation, or on ground computers performing complex GNC algorithms with commands relayed to the vehicle, limits the environments that can be effectively explored. This section describes a navigation system that combines a traditional EKF-based navigation algorithm with IMU measurements and SLAM-based position estimates in lieu of GPS. It utilizes a scanning laser range sensor and sonar altimeter to measure position relative to the environment, and a SLAM algorithm to estimate vehicle position in the local inertial reference frame. A novel method is presented below for determining the uncertainty of pose estimates related to the topology of the environment.

A laser scanner model and simulation environment were developed and used to test different SLAM algorithms prior to using actual hardware. A variety of SLAM algorithm implementations are available for free use at the web site OpenSLAM.org. The algorithm used for this research, called CoreSLAM,[16] was chosen primarily because it is simple, easy to implement, and it uses integer math where possible to improve computational speed.[17] There are two main parts to any SLAM routine. The first task is to measure distance to obstacles or landmarks in the environment, and to map them given the vehicle's position and orientation (i.e. mapping). The second task is to determine the best estimate of the vehicle's position and orientation based on the latest scan (or series of scans) given a stored map (i.e. localization). The mapping and localization tasks are performed together to maintain the most current map and position estimate. As discussed in [15], the robotics and computer science communities place much emphasis on keeping track of vehicle motion and solving chains of pose constraints between differen locations to make corrections to a global map. For basic indoor navigation, however, it was determined that this level of accuracy in a global map is not required, nor even desired if it comes at too steep a computational burden. As a result, the algorithm discussed here is primarily designed for localization with respect to the immediate environment, not to building and maintaining a highly accurate global map. As such, the CoreSLAM mapping routine as implemented here does not detect or correct errors in past observations.

An important aspect of incorporating any measurement into the navigation state estimate is the uncertainty associated with the measurement. The CoreSLAM routine provides a 2-D position measurement and a heading measurement, together called the vehicle *pose*. CoreSLAM and other SLAM algorithms can provide the "fit quality" of the scan measurement compared to the stored map. However, this and all similar algorithms fail to identify the pose uncertainty due to the *structure* of the environment. For example, a vehicle in a long, straight hallway may have a good estimate of its position with respect to the walls on each side, but have very little certainty of its position along the length of the hallway. Similarly, a vehicle

in the middle of a circular room might have a good position estimate, but estimating heading from scan data would be impossible. In these situations, a scan-matching algorithm may produce a fit that has very little error, while at the same time providing very little information in certain directions.[15] This Dilution of Precision (DOP) problem results in a pose estimate that quickly becomes overconfident in directions where few features are observed. To solve this problem, a novel method was developed to identify the uncertainty in the x, y, and θ directions based on the orientation and distance to line segments extracted from the scan data.

A. Dilution of Precision

A typical scanning laser rangefinder performs its measurements by panning a laser across the environment and sampling the return at fixed angular increments. The result is a series of 1-D range readings recorded at consecutive, known angles. The error associated with each range measurement is a function of many parameters, but only the effect of range to target can be estimated for an individual measurement during real-time operation. As a result, it is common practice to estimate the range error as a function of the measured range, neglecting all other sources of error.[18,19] However, if the local topology as indicated by the adjacent measurements is also considered, more information about the measurement uncertainty can be gleaned.

A similar problem exists for range measurements calculated using GPS signals, where the effect of satellite position with respect to the receiver has an impact on the quality of the receiver position estimate. When the satellites used to estimate position are all located in a similar direction, the resulting navigation solution is less accurate. Similarly, the topology of the local environment measured by the laser scanner affects the accuracy of the pose estimate created when matching scans against the map. In essence, vehicle position information is most accurate when measured perpendicular to the local environment. As such, a long featureless wall or hallway only provides position information perpendicular to the walls, and no information parallel to the walls. Thus, any position estimate using scans in this environment will show a strong correlation between the longitudinal and lateral estimates. This correlation is not detected by scan matching routines alone–the shape of the environment must be analyzed. Likewise, a good heading estimate requires that range readings be different in different directions such that any change in heading can be detected. Hence, straight walls provide a good basis for measuring heading, while concave curved surfaces do not.

The information contained in a laser scan can be calculated by summing the information provided by each measurement to form the *information matrix*. An inverse form of the Kalman filter, called the *information filter*, utilizes the information matrix as defined in Equation 15.[20] In this formulation, the *information* contained in each measurement is the inverse of the variance of the measurement, transformed by a measurement matrix, $\mathbf{H}$. In this case, the measurement matrix projects the variance, which is in the range direction, onto the vector normal to the surface. Then, the (x,y) components are calculated to get the information in each direction. For the heading component, note that the range measurement has more heading information when the surface is close to parallel and farther away from the laser source, with the heading in the body frame relative to walls (ψ) being the same as the direction to the normal (ϕ). The geometry of the problem is shown in Figure 6, with the measurement matrix defined in Equation 16.

$$\mathbf{I} \triangleq \mathbf{H}^T {\mathbf{\Sigma_r^2}}^{-1} \mathbf{H} = \mathbf{R}^{-1} \tag{15}$$

$$\mathbf{H} = \begin{bmatrix} \cos(\phi)\cos(\phi_1 - \theta_1) & \sin(\phi_1)\cos(\phi_1 - \theta_1) & r_1 \sin(\phi_1 - \theta_1) \\ \vdots & \vdots & \vdots \\ \cos(\phi)\cos(\phi_m - \theta_m) & \sin(\phi_m)\cos(\phi_m - \theta_m) & r_m \sin(\phi_m - \theta_m) \end{bmatrix} \tag{16}$$

$$\mathbf{\Sigma_r^2} = diag(\sigma_{r_1}^2, \sigma_{r_1}^2, \sigma_{r_1}^2, \ldots, \sigma_{r_i}^2, \sigma_{r_i}^2, \sigma_{r_i}^2, \ldots, \sigma_{r_m}^2, \sigma_{r_m}^2, \sigma_{r_m}^2) \tag{17}$$

The surface normal at each point is calculated by fitting a line through the measurement point and the two points on either side of it. The angle to the normal (ϕ_i) and the point coordinates (r_i,θ_i) are calculated for each range measurement. The result can be calculated as a sum of terms over the number of points in the correspondence set, m. The equation for calculating the information matrix is given by Equation 18,

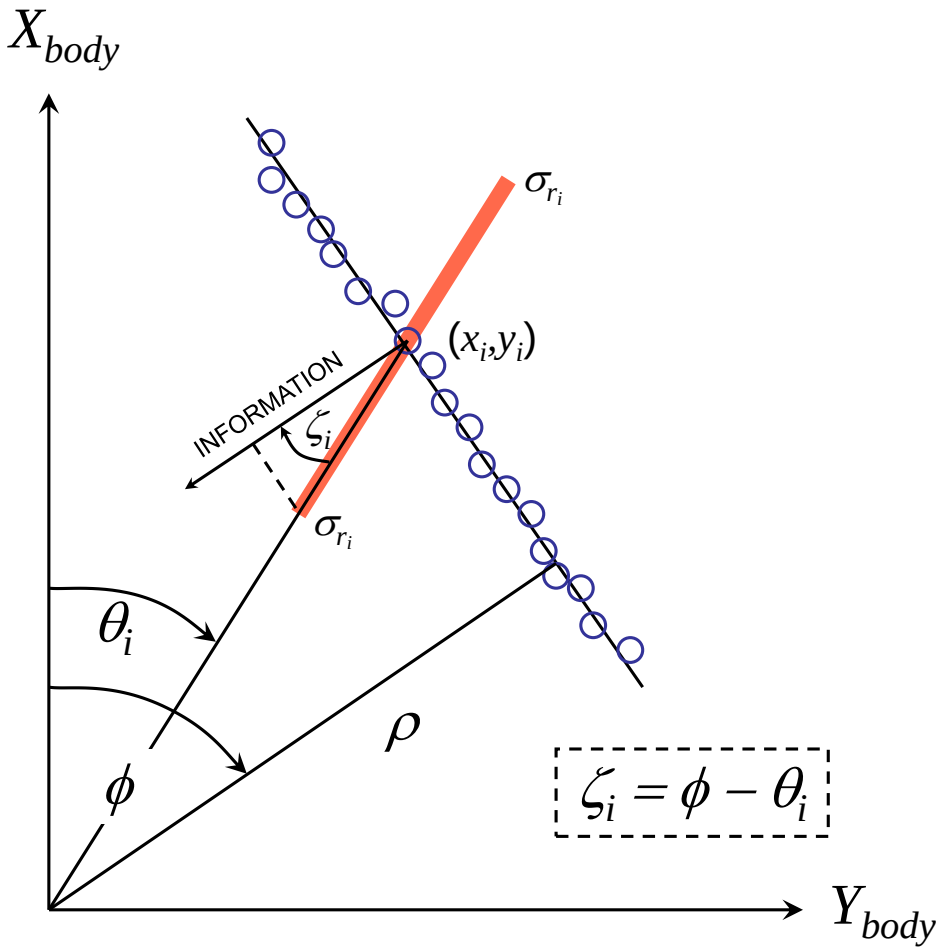


Figure 6. The information available from a laser scan is in the direction perpendicular to the local topology. The range error at each measurement angle is projected onto the information vector direction.

where the summation terms are defined in Equation 19. Inverting the 3×3 information matrix produces the covariance matrix for the $(\Delta x, \Delta y, \Delta\theta)$ pose error measurement, $\mathbf{R}$, which is used to update the EKF state estimate.

$$\mathbf{R} = \mathbf{I}^{-1} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{x\psi} \\ \Sigma_{xy} & \Sigma_{yy} & \Sigma_{y\psi} \\ \Sigma_{x\psi} & \Sigma_{y\psi} & \Sigma_{\psi\psi} \end{bmatrix}^{-1} \tag{18}$$

$$\begin{aligned} \Sigma_{xx} &\triangleq \sum_{i=1}^{m} \frac{\cos^2(\phi)\cos^2(\phi-\theta_i)}{\sigma_{r_i}^2} & \Sigma_{xy} &\triangleq \sum_{i=1}^{m} \frac{\cos(\phi)\sin(\phi)\cos^2(\phi-\theta_i)}{\sigma_{r_i}^2} \\ \Sigma_{yy} &\triangleq \sum_{i=1}^{m} \frac{\sin^2(\phi)\cos^2(\phi-\theta_i)}{\sigma_{r_i}^2} & \Sigma_{x\psi} &\triangleq \sum_{i=1}^{m} \frac{\rho\cos(\phi)\cos(\phi-\theta_i)\sin(\phi-\theta_i)}{\sigma_{r_i}^2} \\ \Sigma_{\psi\psi} &\triangleq \sum_{i=1}^{m} \frac{\rho^2\sin^2(\phi-\theta_i)}{\sigma_{r_i}^2} & \Sigma_{y\psi} &\triangleq \sum_{i=1}^{m} \frac{\rho\sin(\phi)\cos(\phi-\theta_i)\sin(\phi-\theta_i)}{\sigma_{r_i}^2} \end{aligned} \tag{19}$$

While the above form preserves at least some geometric insight into the problem, the speed of the algorithm can be significantly improved in actual implementation by noting the identities in Equations 20 and 22 below. Substitution, and a little careful algebra, produces the form shown in Equation 23, thus enabling the calculation of the information matrix without having to resort to trigonometric operations. Figures 7-9 show the error ellipses associated with the pose measurement covariance calculated using the algorithm presented here using scan data collected during the experiment described at the end of this section. In the figures, the error is normalized using the sensor standard deviation and scaled up as indicated in the plots to improve visualization of the shape and relative size of the measurement uncertainty. The 1-σ position uncertainty is shown by an ellipse, while the 1-σ heading uncertainty is shown by a circle. In comparing the different scenarios, the relative size of the circle indicates the relative uncertainty in the heading estimates, while the size and shape of the error ellipse indicates the relative position uncertainty in the pose estimation.

$$\rho = \left(A^2 + B^2\right)^{-1/2} \tag{20}$$

$$\phi = \arctan\left(\frac{B}{A}\right) \tag{21}$$

$$\begin{aligned} \cos\phi &= \rho A & \cos(\phi-\theta_i) &= \cos\phi\cos\theta_i + \sin\phi\sin\theta_i \\ \sin\phi &= \rho B & \sin(\phi-\theta_i) &= \sin\phi\cos\theta_i - \cos\phi\sin\theta_i \\ \cos\theta_i &= \tfrac{x_i}{r_i} & \sin\theta_i &= \tfrac{y_i}{r_i} \end{aligned} \tag{22}$$

$$\begin{aligned} \Sigma_{xx} &\triangleq \sum_{i=1}^{m} \frac{A^2[A^2x_i^2 + 2ABx_iy_i + B^2y_i^2]}{C_i} & \Sigma_{xy} &\triangleq \sum_{i=1}^{m} \frac{AB[A^2x_i^2 + 2ABx_iy_i + B^2y_i^2]}{C_i} \\ \Sigma_{yy} &\triangleq \sum_{i=1}^{m} \frac{B^2[A^2x_i^2 + 2ABx_iy_i + B^2y_i^2]}{C_i} & \Sigma_{x\psi} &\triangleq \sum_{i=1}^{m} \frac{A[AB(x_i^2 - y_i^2) - x_iy_i(A^2 - B^2)]}{C_i} \\ \Sigma_{\psi\psi} &\triangleq \sum_{i=1}^{m} \frac{[B^2x_i^2 - 2ABx_iy_i + A^2y_i^2]}{C_i} & \Sigma_{y\psi} &\triangleq \sum_{i=1}^{m} \frac{B[AB(x_i^2 - y_i^2) - x_iy_i(A^2 - B^2)]}{C_i} \end{aligned} \tag{23}$$

$$C_i \triangleq (A^2 + B^2)^2 r_i^2 \sigma_{r_i}^2;$$

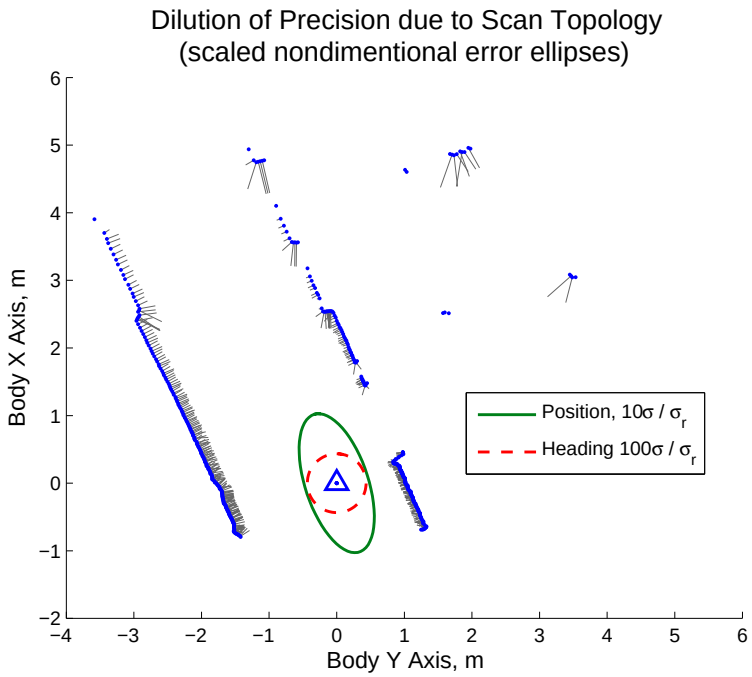


Figure 7. In a hallway, uncertainty is greater in the direction of the hallway.

B. CoreSLAM Implementation

In CoreSLAM, estimated vehicle state information is used to align each new scan with the evolving map. CoreSLAM maintains a map that consists of a two-dimensional array containing integer values ranging from 0 to 65535. The map is initialized to a middle value of 32768, and values are adjusted as each new scan is processed to reflect the evolving map. In order to easily visualize the map, the values are scaled to the range 0-255 and displayed on a one-to-one scale 8-bit gray scale image. As observations are made, areas where obstacles are detected are darkened, and areas that are clear of obstacles increase in brightness. The map image thus represents an occupancy grid, with color value displaying the probability that a square is occupied. As more areas are explored, the observed obstacles are shown by darkened areas on the map, while clear areas are displayed by lighter colored pixels.

Once the map is initialized, the algorithm continuously monitors and incorporates new scan data into the map. The scan data is transmitted from the sensor as a vector of range measurements, where the angle increments counterclockwise. CoreSLAM requires the scan data to be in a Cartesian coordinate system, so the scan data is first converted from polar coordinates. Next, the position estimate is updated if desired. The vehicle always maintains an estimate of its position in the navigation state vector, and this estimate is used as a starting point for the localization routine. A scan registration algorithm is used to determine the most likely vehicle position and orientation (pose) for a given scan. This is accomplished by performing a Monte Carlo search of different poses nearest the current pose estimate, and evaluating the latest scan at the new pose to see how closely it matches the most current map. The current pose estimate is updated by using the vehicle's EKF navigation solution to provide a more accurate starting point for the Monte Carlo search. If the scan matching routine can find a better pose to match the current scan with the most recent map, the current pose is updated to the new pose. In [17], the measure of how well a particular scan matches the latest map is described as the "distance" between the scan and the map. The Monte Carlo routine calculates the distance between the scan and the map and returns the pose that minimizes this distance. Next, the scan data is incorporated into the map using the updated pose. Figure 10 shows a flow chart that illustrates how the CoreSLAM mapping and localization algorithms are implemented. Figure 11(a) shows a simulated building interior being explored by a helicopter with a scanning laser rangefinder. Figure 11(b) shows the

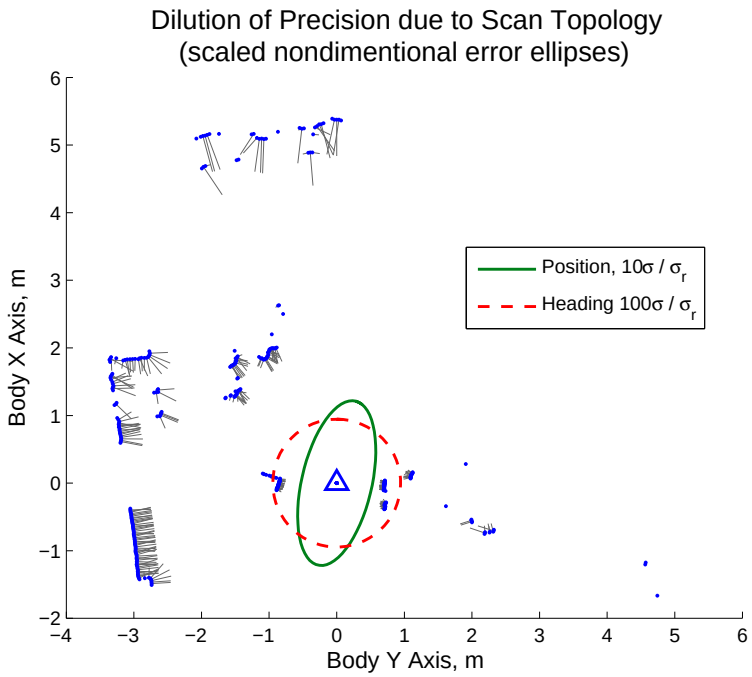


Figure 8. If many surface normals point toward the vehicle, the heading uncertainty is greater.

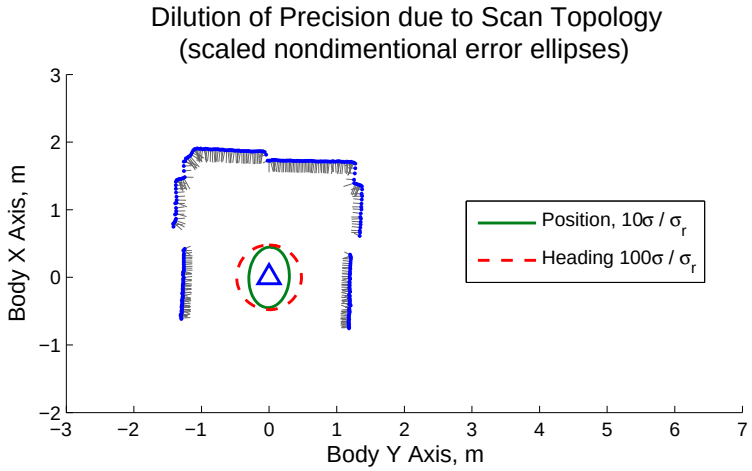


Figure 9. An environment with walls on several sides reduces uncertainty in the pose measurement.

map generated during a simulated flight.

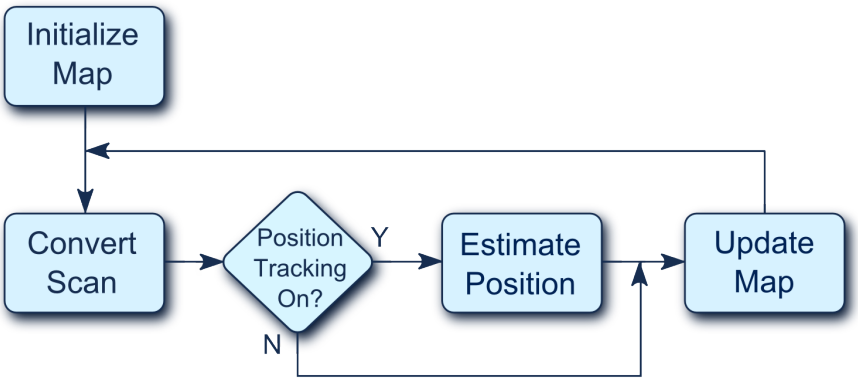


Figure 10. This flowchart shows the CoreSLAM algorithm as implemented in simulation.

For this research, the CoreSLAM algorithm was modified and improved in two important ways.[15] First, the position covariance matrix of the navigation solution was used as an input to the map update function. In the original algorithm, a user-defined constant value was used to create the Gaussian uncertainty on obstacle locations. To prevent unrealistic confidence in the map, the actual sensor range uncertainty (which is a function of range detected) was added to the vehicle's position uncertainty. The map update algorithm was then modified to use this total uncertainty when assimilating scan data into the map. A second improvement made to the CoreSLAM algorithm was to use the vehicle's state estimate and covariance as inputs to the Monte Carlo search. Thus, the random search initial conditions and search scope were significantly improved over the original algorithm by providing a good starting point for the search, resulting in better match performance.

C. Extended Kalman Filter State Estimation

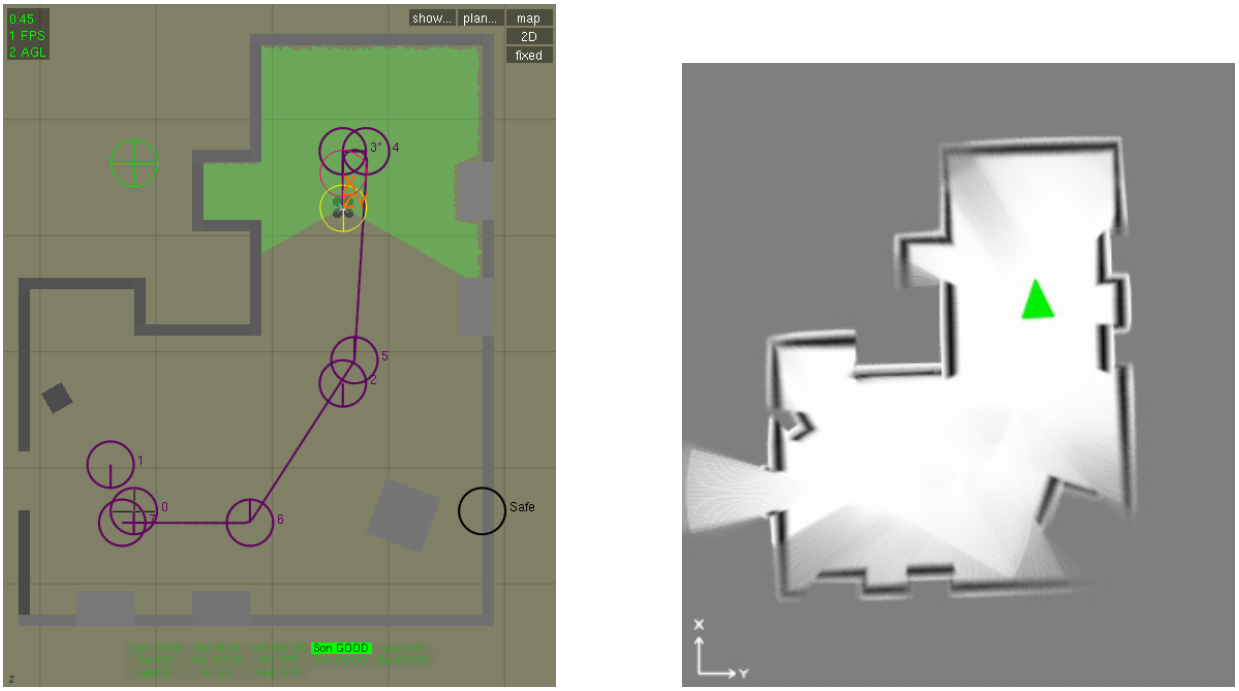
Vehicle state estimation for the navigation algorithm is based on the EKF design developed at the GT UAV Research Facility.[2] In this implementation, the IMU measurements are incorporated at a fixed rate of 100Hz. In lieu of GPS position measurements and magnetometer heading measurements, the SLAM pose estimate and sonar altitude measurements are used, with updates at 10Hz and 20Hz respectively. The navigation filter estimates 15 vehicle states, $\hat{\mathbf{x}}(\mathbf{t})$, which are propagated using the linearized vehicle dynamics and updated using the sensor measurements with the appropriate covariance $\mathbf{R}_k$ and measurement matrix $\mathbf{H}_k$. The state and covariance propagation, as well as updates using the IMU measurements, are handled by the existing navigation software. Measurement updates from the SLAM pose estimation and sonar altimeter are incorporated by calling an external update function that takes as its arguments the measurement residual (sometimes called the "innovation"), the measurement matrix, and the covariance of the measurements. The measurement update equations are show below in Equations 24-26.[21]

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\big(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)\big) \tag{24}$$

$$\mathbf{P}_k^+ = \big(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\big)\mathbf{P}_k^- \tag{25}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-)\big(\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-) + \mathbf{R}_k\big)^{-1} \tag{26}$$

As shown in Equation 27, the state estimate includes the vehicle angle error, position, velocity, and biases of the IMU accelerometers and gyros. The process model, given by Equations 28-31 below, is a set of nonlinear differential equations describing the vehicle motion as described in detail in [2, 22]. The residual of the SLAM pose measurements (Equation 32) is the pose error (Δx, Δy, $\Delta\theta$), which is expressed in the inertial frame. The altitude residual (Equation 33) is simply the difference between the sonar measurement and the altitude estimate. It is assumed that the vehicle attitude does not affect the altitude measurement due to



(a) Mapping a simulated environment

(b) Map generated during simulation

Figure 11. The map maintained by the CoreSLAM routine represents an occupancy grid, where the value of each pixel in the image represents the likelihood that a particular grid square is occupied. Here, lighter colors represent free space, while darker colors represent obstacles and medium gray areas are unexplored. Areas with higher contrast represent greater certainty due to longer observation periods during the flight. The green triangle represents the vehicle's estimated position and heading.

the properties of the sensor, as described in [23]. The state and covariance updates occur at different times due to the different measurement rates of the SLAM navigation and the sonar. The sonar measurement is uncorrelated to other measurements, so its experimentally determined variance is used directly in the Kalman gain (26), with a measurement matrix equal to unity. The covariance of the SLAM pose measurement, calculated by inverting the information matrix shown in Equation 15, is in the body frame. Hence, it must be transformed using the direction cosine matrix that rotates body frame measurements into the inertial frame.

$$\hat{\mathbf{x}}(\mathbf{t}) = [\phi_{err}, \theta_{err}, \psi_{err}, x_{pos}, y_{pos}, z_{pos}, u, v, w, b_{ax}, b_{ay}, b_{az}, b_{\omega x}, b_{\omega y}, b_{\omega z}]^T \tag{27}$$

$$\dot{\hat{\mathbf{x}}} = \hat{\mathbf{v}} \tag{28}$$

$$\dot{\hat{\mathbf{v}}} = a(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})) \tag{29}$$

$$\dot{\hat{\mathbf{b}}}_{\mathbf{a}} = 0 \tag{30}$$

$$\dot{\hat{\mathbf{b}}}_{\omega} = 0 \tag{31}$$

$$\left(\mathbf{z}_{SLAM_k} - h(\hat{\mathbf{x}}_k^-)\right) = [\Delta x, \Delta y, \Delta\theta]^T \tag{32}$$

$$\left(\mathbf{z}_{sonar_k} - h(\hat{\mathbf{x}}_k^-)\right) = (sonar\_measurement_k - EKF\_alt\_estimate_k) \tag{33}$$

VI. Control Algorithm

A. Stability Augmentation System (SAS)

The Quadrotor platform is inherently unstable, that is, without control inputs, the platform would enter an uncontrolled drift in velocity and angular rates and collide with the ground or nearby obstacles. Quadrotors are also known to be notoriously hard to control even for human pilots, particularly because the relationship between thrust and stick deflection is nonlinear and since attitude is coupled heavily with velocity. Hence, it is desirable to integrate rate damping on all the angular rate axis to aid the pilot in controlling the Quadrotor. Let $\hat{p}$, $\hat{q}$, and $\hat{r}$ denote the gyroscope measurements of the Quadrotor roll, pitch, and yaw rates, and δ_{p_p}, δ_{q_p}, and δ_{r_p} denote the pilot roll, pitch, and yaw stick deflections, then the actual stick deflection commands are assigned using the following proportional control logic:

$$\delta_p = \delta_{p_p} - K_p\hat{p}, \tag{34}$$

$$\delta_q = \delta_{q_p} - K_q\hat{q}, \tag{35}$$

$$\delta_r = \delta_{r_p} - K_r\hat{r}. \tag{36}$$

In the above equation, K_p, K_q, and K_r denote the linear gains chosen to provide appropriate rate damping.

B. Attitude and Position Controller

The complexity of the control system depends not only on the quantities being controlled, but also on the dynamics of the system itself. Unlike ground vehicles, unstable air vehicles are susceptible to oscillation and divergent flight when the control system is not properly tuned. Even for stable flying vehicles, coupling between lateral and longitudinal motion, as well as aerodynamic interaction with the environment, must be considered. The control architecture used by the GTAR 2010 team leverages the proven Model Reference Adaptive Control architecture developed for control of VTOL UAS throughout their flight envelop by Georgia Tech UAV Research Facility.[24–26] In this architecture, a position control loop generates a velocity command, a velocity control loop generates an attitude command, and an attitude control loop generates servo commands to stabilize the vehicle by controlling the angular rate. Kannan has shown that such nested and cascaded control loop architecture with actuator saturation can indeed be used to control VTOL UAS.[25]

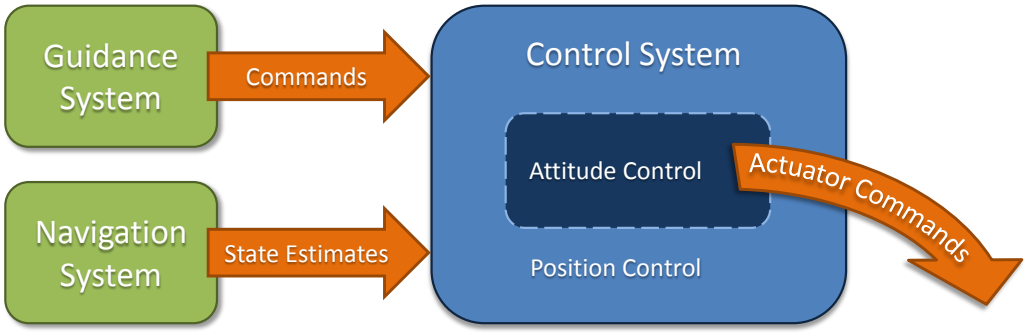


Figure 12. A common control architecture for unstable rotorcraft utilizes a set of nested loops to control attitude, velocity, and ultimately position by generating actuators commands. The guidance system provides position, velocity, and attitude commands, and the navigation system provides an estimate of the system states.

This system of nested control loops (see Figure 12) requires that the vehicle maintain an estimate of its position, velocity, attitude, and angular rate, with the addition of an external inertial position reference if position control is to be ultimately achieved. Following is a brief description of the implemented control architecture (see references [24, 25, 27, 28] for further details).

Let $x(t) \in \Re^n$ denote the state vector, let $\delta \in \Re^m$ denote the control input, then the dynamics of the

Quadrotor can be generically represented as:

$$\dot{x} = f(x(t), \delta(t)). \tag{37}$$

It is assumed that the exact model, Eq.(37) is not available, may have changed, or is not sufficiently accurate and introduce an approximate inversion model $\hat{f}(x, \delta_{cmd})$ which can be inverted to determine the control input: $\delta = \hat{f}^{-1}(x, \nu)$. Here ν is the pseudo control input, which represents the desired model output $\dot{x}$ and is expected to be approximately achieved by δ. This approximation results in a model error $\Delta(x) = f(x) - \hat{f}(x)$.[24,27,28]

A second order reference model is selected to characterizes the desired response of the system:

$$\dot{x}_{rm} = f_{rm}(x_{rm}, r(t)), \tag{38}$$

Where $f_{rm}(x_{rm}(t), r(t))$ denote the reference model dynamics and $r(t)$ denotes a bounded and piecewise continuous external command. A tracking control law consisting of a linear feedback part $u_{pd} = Kx$, a linear feed-forward part $u_{crm} = \dot{x}_{rm}$, and an adaptive part $u_{ad}(x)$ is proposed to have the following form:

$$u = u_{crm} + u_{pd} - u_{ad}. \tag{39}$$

Defining the tracking error e as $e(t) = x_{rm}(t) - x(t)$, then, letting $A = -K$ the tracking error dynamics are found to be:[27]

$$\dot{e} = Ae + [\nu_{ad}(x) - \Delta(x)]. \tag{40}$$

The baseline full state feedback controller $u_{pd} = Kx$ is assumed to be designed such that A is a Hurwitz matrix, hence for any positive definite matrix $Q \in \Re^{n\times n}$, a positive definite solution $P \in \Re^{n\times n}$ exists to the Lyapunov equation. This architecture allows for an optional adaptive element (whose output is denoted by u_{ad}) to be incorporated for mitigating model errors and reducing steady-state tracking error. The adaptive part of the control law is represented using a Single Hidden Layer Neural Network (NN) or a Radial Basis Function NN:

$$u_{ad}(x) = \hat{W}^T \sigma(x), \tag{41}$$

Using theory and extensive flight testing, it has been demonstrated that the stability of the presented MRAC architecture for trajectory tracking control of VTOL UAS[24,25] with adaptive laws similar to:

$$\dot{\hat{W}} = -\Phi(x) e^T P B \Gamma_W - \kappa \|e\| W \tag{42}$$

where Γ_W is a positive definite learning rate matrix, and κ denotes the gain for an e-modification term.[29] In the presented control architecture, the output of the adaptive element can be replaced by an integrator if desired. This reduces the control logic to Proportional-Derivative-Integral type.

VII. Results

A. Simulation Environment

The Georgia Tech UAV Simulation Tool (GUST) framework was utilized for developing a simulation[30] of the vehicle, sensors, and environment. The simulation is designed to reduce development time through risk-free testing of guidance, navigation, and control routines. The simulation includes modeling of uncertainties such as gusts, modeling of indoor environments, simulation of complex vehicle dynamics, and elaborate emulation of all sensors and their noise properties. The key feature of this simulation environment is that onboard code can be directly tested in the simulation, allowing seamless integration of Software-In-The-Loop (SITL) and Hardware-In-The-Loop (HITL) simulation. Figure 13 depicts a screen-shot of the vehicle in simulation.

B. Integrated Guidance Navigation and Control Results

Figure 14 shows a sequence of screen-captures from a simulated flight of the developed integrated guidance navigation and control strategy. The vehicle, it's trajectory, commanded waypoint, laser scan, active SRT, and the building are shown on Figure 14. The results confirm that the vehicle is able to sense its surrounding, form a feasible six degree of freedom navigation solution, and autonomously avoid obstacles while creating

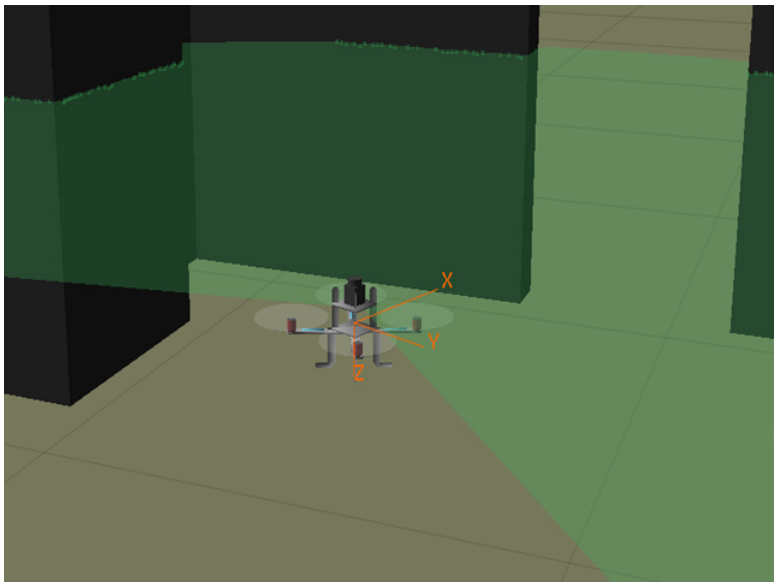


Figure 13. Quadrotor Performing Wall Following in Simulation

SRT and explores the building. In Figure 14(c), the quadrotor is able to get pass through a small gap on the northwest corner without collision. The quadrotor then explores towards the east of the building, attempting to do an exhaustive search of the building. The detailed results can be found in[31]

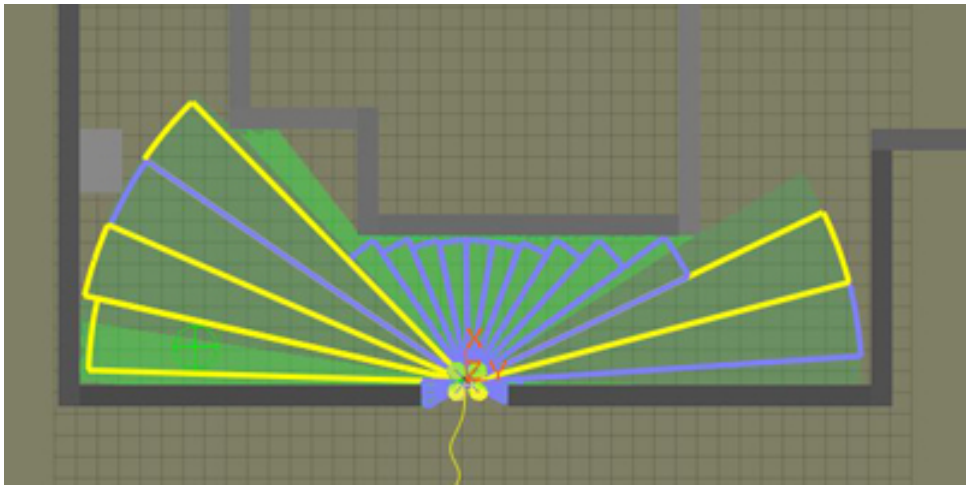
C. Flight Test Results

D. Flight Test Results

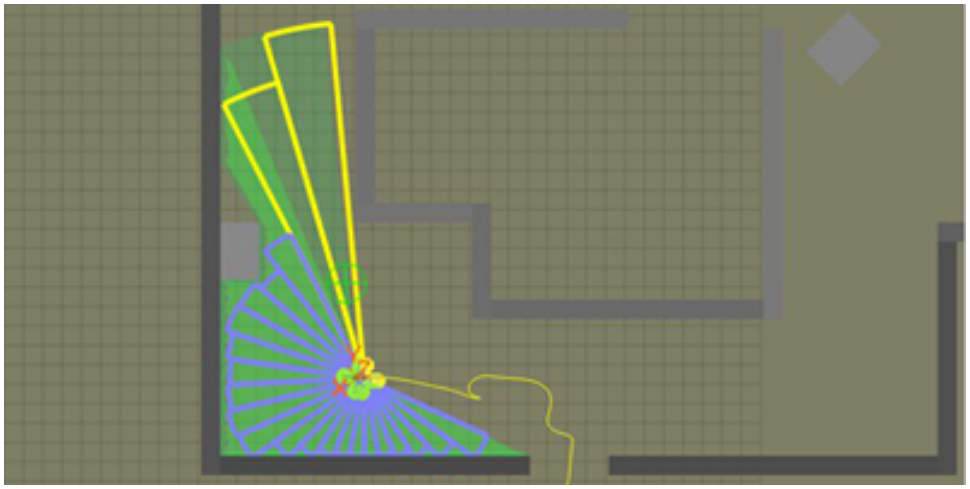
In this section we present flight test results of the GTQ exploring an indoor cluttered environment fully autonomously without any external sensing aids (such as GPS). The onboard GNC algorithm does not assume any a-priori knowledge of the indoor environment. Navigation is performed by solving a SLAM problem online using techniques presented in Section V. Guidance is achieved by frontier-guidance type method coupled with wall-following guidance as described in Section IV, and control is achieved using the control architecture described in Section VI. The flight test begins with the aircraft hovering at about 2.8 ft above the ground. The onboard guidance logic then commands waypoints that take the aircraft towards unexplored frontiers, the onboard navigation logic provides the aircraft with its pose information and simultaneously builds a map of the environment in real-time. The map information is fed back into the guidance logic to explore new frontiers. Note that all computation, including SLAM is performed online using the avionics described in Section III. The GNC algorithms were optimized to execute completely onboard the embedded computer (Gumstix Overo Fire) used by trading-off map accuracy with guaranteeing a reliable instantaneous position fix for pose estimation. This trade-off results in a slight skew in the onboard generated map (Figure 15). However, the attitude estimation was sufficiently accurate for good attitude control and the position accuracy was found acceptable for exploring indoor areas reliably.

VIII. Conclusion

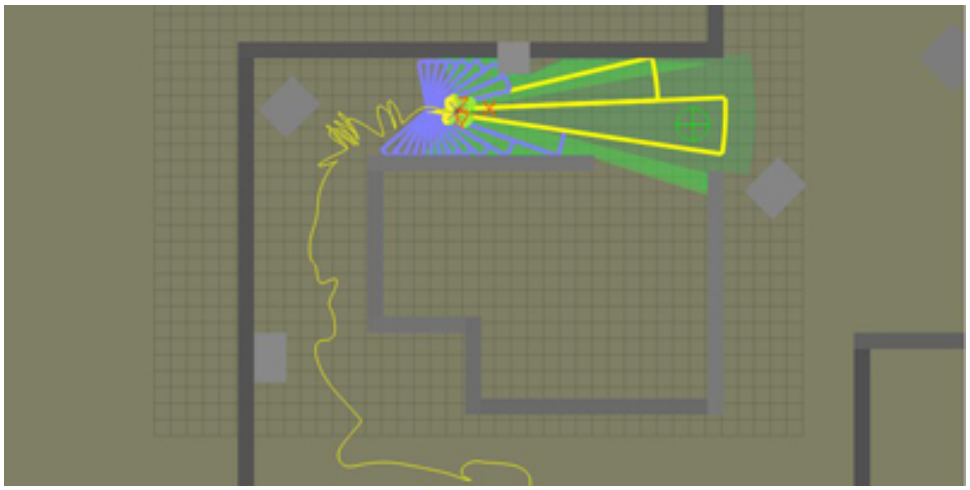
This paper presented the details of a Quadrotor Unmanned Aerial Vehicle intended for autonomous exploration of indoor areas. The vehicle uses an off-the-shelf platform equipped with off-the-shelf avionics and sensor packages, with custom flight software. Information from a scanning laser range sensor, inertial measurement unit, and a altitude measurement sonar are fused to form an elaborate navigation solution using Simultaneous Localization and Mapping methods. An important feature of this navigation architecture is that it does not rely on any external navigational aid, such as GPS. A frontier-based exhaustive search is used for exploring unknown indoor environments using the laser scan data by placing command waypoints on the map generated by the SLAM routine. Since observable features in the environment are necessary for



(a)

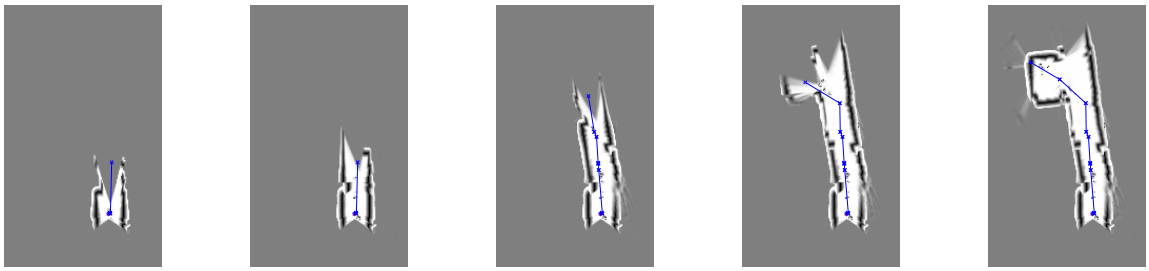


(b)



(c)

Figure 14. Simulation of Exploration of Unknown Indoor Environment



(a) Time = 19 s (b) Time = 40 s (c) Time = 61 s (d) Time = 92 s (e) Time = 124 s

Figure 15. The GTQ autonomously explores an unknown cluttered indoor environment without any external sensing or computational aids. The figures show the map of the unknown indoor environment generated by the onboard navigation algorithm as the GTQ explores the indoor environment. Stars mark the waypoints commanded by the guidance strategy, the guidance strategy ensures the GTQ explores unexplored areas of the indoor environment. The pictured area is approximately 50 by 80 feet. Note that all computation, including solving the SLAM problem, is performed onboard.

an accurate SLAM solution, the guidance algorithm is coupled with the navigation algorithm to ensure the vehicle approaches the frontier-based waypoints through a trajectory that is close to walls and other indoor structures, while maintaining a safe operating distance. A cascaded inner-outer loop control architecture is utilized, which relates stick commands to attitude commands and attitude commands to velocity commands. The control architecture employs an optional adaptive element which can be used to mitigate modeling error and other system uncertainties. The control system also uses a linear Stability Augmentation System that uses rate feedback to dampen the vehicle angular rate response.

An elaborate simulation model of the vehicle has been developed and the navigation and control algorithms have been validated in simulation and several times in flight. These results establish the feasibility of the proposed approach to develop a completely self-contained miniature unmanned aerial system capable of autonomously exploring indoor areas.

Acknowledgments

The authors wish to thank the members and friends of the Georgia Tech Aerial Robotics team, including Sushaku Yamaura, Jeong Hur, Dr. Suresh Kannan, Nimrod Rooz, and Dr. Erwan Salaün.

References

[1]Kayton, M. and Fried, W. R., *Avionics Navigation Systems*, John Wiley and Sons, 1997.

[2]Christophersen, H. B., Pickell, W. R., Neidoefer, J. C., Koller, A. A., Kannan, S. K., and Johnson, E. N., "A compact Guidance, Navigation, and Control System for Unmanned Aerial Vehicles," *Journal of Aerospace Computing, Information, and Communication*, Vol. 3, May 2006.

[3]Wendel, J., Maier, A., Metzger, J., and Trommer, G. F., "Comparison of Extended and Sigma-Point Kalman Filters for Tightly Coupled GPS/INS Integration," *AIAA Guidance Navigation and Control Conference*, San Francisco, CA, 2005.

[4]Bouabdallah, S., Noth, A., and R., S., "PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor," *Proc. of The IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[5]Portlock, J. and Cubero, S., "Dynamics and Control of a VTOL quad-thrustaerial robot," *Mechatronics and Machine Vision in Practice*, edited by J. Billingsley and R. Bradbeer, 2008.

[6]Guo, W. and Horn, J., "Modeling and simulation for the development of a quad-rotor UAV capable of indoor flight," *Modeling and Simulation Technologies Conference and Exhibit*, 2006.

[7]Johnson, E. and Turbe, M., "Modeling, Control, and Flight Testing of a Small Ducted Fan Aircraft," *Journal of Guidance Control and Dynamics*, Vol. 29, No. 4, July/August 2006, pp. 769–779.

[8]Achtelik, M., Bachrach, A., He, R., Prentice, S., and Roy, N., "Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments," *Proc. of the 1st Symposium on Indoor Flight, International Aerial Robotics Competition*, 2009.

[9]Sobers, D. M., Yamaura, S., and Johnson, E. N., "Laser-Aided Inertial Navigation for Self-Contained Autonomous Indoor Flight," *Proc. AIAA Guidance, Navigation, and Control Conference (GNC'10)*, Toronto, Ontario, Canada, Aug. 2010.

[10]Yamauchi, B., "A Frontier-Based Approach for Autonomous Exploration," *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA97)*, Monterey, CA, July 1997, pp. 146–151.

[11]Gonzalez-Banos, H. and Latombe, J.-C., "Navigation Strategies for Exploring Indoor Environments," *Int. J. Robotics Research*, Vol. 21, No. 10, 2002, pp. 829–848.

[12]Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F., "An Experiment in Integrated Exploration," *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS02)*, 2002.

[13]Freda, L. and Oriolo, G., "Frontier-Based Probabilistic Strategies for Sensor-Based Exploration," *Proc. IEEE International Conference on Robotics and Automation (ICRA05)*, Barcelona, Spain, April 2005.

[14]Freda, L., Loiudice, F., and Oriolo, G., "A Randomized Method for Integrated Exploration," *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS06)*, Beijing, China, Oct. 2006.

[15]D. Michael Sobers, J., Yamaura, S., and Johnson, E. N., "Laser-Aided Inertial Navigation for Self-Contained Autonomous Indoor Flight," *AIAA Guidance Navigation and Control Conference*, Toronto, Ontario, Canada, August 2010.

[16]Steux, B. and Hamzaoui, O. E., "CoreSLAM on OpenSLAM.org," Jun 2010.

[17]Steux, B. and Hamzaoui, O. E., "CoreSLAM : a SLAM Algorithm in less than 200 lines of C code," Tech. rep., Mines ParisTech, Center for Robotics, Paris, France, 2009.

[18]Arras, K. O. and Siegwart, R. Y., "Feature extraction and scene interpretation for map-based navigation and map building," Vol. 3210, SPIE, 1998, pp. 42–53.

[19]Núñez, P., Vázquez-Martín, R., del Toro, J., Bandera, A., and Sandoval, F., "Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation," *Robotics and Autonomous Systems*, Vol. 56, No. 3, 2008, pp. 247 – 264.

[20]Durrant-Whyte, H., "Equations for the Prediction Stage of the Information Filter," .

[21]Gelb, A., Joseph F. Kasper, J., Raymond A. Nash, J., Price, C. F., and Arthur A. Sutherland, J., *Applied Optimal Estimation*, The M.I.T. Press, 1974.

[22]Johnson, E. N., Turbe, M., Wu, A., Kannan, S. K., and Neidhoefer, J. C., "Flight Test Results of Autonomous Fixed-Wing Transition to and from Stationary Hover," *Journal of Guidance Control and Dynamics*, Vol. 31, No. 2, March 2008, pp. 358–370.

[23]Sobers, D. M., Chowdhary, G., and Johnson, E. N., "Indoor Navigation for Unmanned Aerial Vehicles," *AIAA Guidance Navigation and Control Conference*, Georgia Institute of Technology, Chicago, Illinois, August 2009, p. 29.

[24]Johnson, E. and Kannan, S., "Adaptive Trajectory Control for Autonomous Helicopters," *Journal of Guidance Control and Dynamics*, Vol. 28, No. 3, May 2005, pp. 524–538.

[25]Kannan, S. K., *Adaptive Control of Systems in Cascade with Saturation*, Ph.D. thesis, Georgia Institute of Technology, Atlanta Ga, 2005.

[26]Chowdhary, G. and Johnson, E., "Flight Test Validation of Long Term Learning Adaptive Flight Controller," *Proceedings of the AIAA GNC Conference, held at Chicago, IL*, 2009.

[27]Johnson, E. N., *Limited Authority Adaptive Flight Control*, Ph.D. thesis, Georgia Institute of Technology, Atlanta Ga, 2000.

[28]Chowdhary, G. V. and Johnson, E. N., "Theory and Flight Test Validation of Long Term Learning Adaptive Flight Controller," *Proceedings of the AIAA Guidance Navigation and Control Conference*, Honolulu, HI, 2008.

[29]Narendra, K. and Annaswamy, A., "A New Adaptive Law for Robust Adaptation without Persistent Excitation," *IEEE Transactions on Automatic Control*, Vol. 32, No. 2, February 1987, pp. 134–145.

[30]Johnson, E. N. and Schrage, D. P., "System Integration and Operation of a Research Unmanned Aerial Vehicle," *AIAA Journal of Aerospace Computing, Information and Communication*, Vol. 1, No. 1, Jan 2004, pp. 5–18.

[31]Pravitra, C., Chowdhary, G., and Johnson, E. N., "A Compact Exploration Strategy for Indoor Flight Vehicle," *American Control Conference*, San Francisco, CA, June 2010.

IX. Appendix

A. Derivation of Moment Equation

Detailed derivation for moment equation is described here. Starting with Euler's law.

$$
\begin{aligned}
M &= {}^{I}\frac{dH}{dt} \\
M &= {}^{I}\frac{d}{dt}[H_b + H_{r1} + H_{r2} + H_{r3} + H_{r4}] \\
M &= {}^{I}\frac{d}{dt}[I\omega] + {}^{I}\frac{d}{dt}\begin{bmatrix} 0 \\ 0 \\ I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4 \end{bmatrix}
\end{aligned}
$$

$$M = I\dot{\omega} + \omega \times I\omega + \begin{bmatrix} 0 \\ 0 \\ I_r\dot{\Omega}_1 - I_r\dot{\Omega}_2 + I_r\dot{\Omega}_3 - I_r\dot{\Omega}_4 \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4 \end{bmatrix}$$

$$M - \begin{bmatrix} q(I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4) \\ -p(I_r\Omega_1 - I_r\Omega_2 + I_r\Omega_3 - I_r\Omega_4) \\ I_r\dot{\Omega}_1 - I_r\dot{\Omega}_2 + I_r\dot{\Omega}_3 - I_r\dot{\Omega}_4 \end{bmatrix} = I\dot{\omega} + \omega \times I\omega$$

where

$$M = \begin{bmatrix} l_y(-\tau_1 - \tau_2 + \tau_3 + \tau_4) \\ l_x(\tau_1 - \tau_2 - \tau_3 + \tau_4) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix}$$